

REMARKS

Applicants acknowledge receipt of the Examiner's Communication mailed February 7, 2001.

By this Amendment, Applicants have amended claims 1, 5, 13, 23, 25 and 59, canceled claims 4 and 60, and added new claim 62. Therefore, claims 1, 2, 5-59, 61 and 62 are presently pending. Applicants respectfully request reconsideration, reexamination and allowance of the application.

On page 2 of the Office Action, the Examiner rejected claim 1 and all claims depending thereon under 35 U.S.C. 112, first paragraph, as containing subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the art that the inventor(s), at the time the application was filed, had possession of the claimed invention. To alleviate this situation, the reference in claim 1 to "a processor and a coprocessor" has been amended to refer only to a coprocessor. Claim 30 is also amended herewith to remove the reference to a processor. Applicants submit that claims 1 and 30 as amended are fully supported by the specification.

The Examiner also rejected claim 61 under 35 U.S.C. 112, first paragraph, and stated that the feature "transferring a third block of processed graphics data from the on-board memory to the main memory while the first block of graphics data is being processed" is not described. Applicants refer the Examiner to page 125, lines 14-16 of the specification, which state that "The DMA engine 1304 preferably transfers data between the memory 28 and the data SRAM 1302 to carry out load and store instructions." Additionally page 126, lines 11-15 indicate that "the graphics accelerator 64 is working on operands and producing outputs for one set of pixels, while the DMA engine 1304 is bringing in operands for the next (future) set of pixel operations, and also the DMA engine 1304 is writing back to memory the results from the previous set of pixel operations." Applicants submit that at least these passages of the specification fully support claim 61.

The Examiner also rejected claims 1, 2, 4, 5, 19, 23-31, 37, 44 and 50-58 under 35 U.S.C., §103(a) as allegedly being unpatentable over Ben-Yoseph et al., U.S. Patent 5,949,439, in view of Gulick et al., U.S. Patent 5,758,177. Applicants have amended claim 1 to recite a graphics accelerator comprising a data memory for storing the graphics

data, the graphics data including pixels, and a coprocessor for performing vector operations on a plurality of components of pixel of the graphics data, wherein the coprocessor processes the plurality of components of the pixel in parallel as elements of a vector. Claim 1 as amended is substantially similar in scope to original claim 4 together with the limitations of original base claim 1. Accordingly, claim 4 is cancelled with this Amendment. The Ben-Yoseph and Gulick references do not render amended claim 1 obvious because, among other reasons, they fail to disclose the claimed combination wherein the coprocessor processes the plurality of components of the pixel in parallel as elements of a vector.

On page 4 of the Office Action, the Examiner asserts that Ben-Yoseph teaches that the coprocessor processes the plurality of components of the pixel in parallel as elements of a vector. Applicants respectfully disagree. Ben-Yoseph discloses a multimedia processor 106 that uses Very Long Instruction Word (VLIW), vector processing, and single-instruction multiple data (SIMD) technology to achieve parallel operation. Col. 3, lns. 35-39. The multimedia processor 106 transmits and receives data simultaneously over a high-speed Rambus DRAM channel I/O bus. Col. 3, lns. 43-45. But to say that the multimedia processor of Ben-Yoseph achieves parallel operation and transmits and receives data simultaneously over a DRAM channel I/O bus is not to say that Ben-Yoseph teaches that the coprocessor processes the plurality of components of a pixel in parallel as elements of a vector, as is claimed in amended claim 1. There is simply no disclosure in Ben-Yoseph of the processor 106 processing a plurality of components of a pixel in parallel as elements of a vector. As neither Ben-Yoseph nor Gulick disclose this element of amended claim 1, Applicants respectfully request allowance of claim 1 and claims dependent thereon.

Claim 23 is amended herewith to be in independent form including the limitations of original base claim 1 and original intervening claim 3. Thus claim 23 includes a direct memory access (DMA) engine for transferring the graphics data between an external memory and the local memory, wherein the DMA engine moves data between the local memory and the external memory while the graphics accelerator is using the memory for its load and store operations. On page 4 of the Office Action, the Examiner asserts that Ben-Yoseph discloses this aspect of the present invention, citing Col. 4, lns. 36-40 and

Col. 4, ln. 65 through Col. 5, ln. 4 of Ben-Yoseph and Col. 6, lns. 42-46 of Gulick. Applicants submit that while Ben-Yoseph and Gulick both disclose using DMA as a means of transferring data between external memory and a processor (or SRAM associated with a processor), nowhere do Ben-Yoseph nor Gulick disclose a DMA engine that moves data between the local memory and the external memory at the same time the graphics accelerator is using the memory for load and store operations. As neither Ben-Yoseph nor Gulick disclose this element of amended claim 23, Applicants respectfully request allowance of claim 23 and claims dependent thereon.

Claim 25 is amended herewith to be in independent form including the limitations of original base claim 1 and original intervening claim 3. Thus claim 25 includes a direct memory access (DMA) engine for transferring the graphics data between an external memory and the local memory, wherein the DMA engine includes a queue to hold a plurality of DMA commands. On page 4 of the Office Action, the Examiner asserts that Ben-Yoseph teaches a DMA engine that includes a queue to hold a plurality of DMA commands. The Examiner contends that this aspect is “inherent by portion of SRAM 146 Col. 5, lns. 50, 62 and/or software queue 144” of Ben-Yoseph. Applicants respectfully disagree. The SRAM 146 of Ben-Yoseph in no way teaches, neither directly nor inherently, a DMA engine having a queue to hold a plurality of DMA commands. It is simply an entirely different animal and is in no way related to what is claimed in claim 25. Likewise, the software queue 144 of Ben-Yoseph in no way teaches, neither directly nor inherently, a DMA engine having a queue to hold a plurality of DMA commands. Applicants point out that claim 25 does not generically claim a software queue. Rather, claim 25 is directed, in part, to a DMA engine having a queue to hold a plurality of DMA commands, which is in no way inherent from the software queue 144 of Ben-Yoseph. As neither Ben-Yoseph nor Gulick disclose this element of amended claim 25, Applicants respectfully request allowance of claim 25 and claims dependent thereon.

Claim 29 is amended herewith to be in independent form including the limitations of original base claim 1 and original intervening claim 3. Thus claim 29 includes a direct memory access (DMA) engine for transferring the graphics data between an external memory and the local memory, wherein the graphics accelerator is working on operands and producing outputs for one set of pixels, while the DMA engine is bringing in

operands for a future set of pixel operations. On page 5 of the Office Action, the Examiner asserts that Ben-Yoseph teaches the graphics accelerator is working on operands and producing outputs for one set of pixels, while the DMA engine is bringing in operands for a future set of pixel operations. The Examiner contends that this aspect is “inherent by the teachings of FIFO type queue or portions of SRAM of Ben-Yoseph et al. and/or software queue 144 and Gulick et al., inherent by the teachings of buffer 234 because this is what the buffer is for.” Applicants respectfully disagree. The SRAM 146 of Ben-Yoseph in no way teaches, neither directly nor inherently, that the graphics accelerator is working on operands and producing outputs for one set of pixels, while the DMA engine is bringing in operands for a future set of pixel operations. In no way does the generic teaching of an SRAM, such as that in Ben-Yoseph, inherently teach a graphics accelerator that works on operands and produces outputs for one set of pixels, while the DMA engine brings in in operands for a future set of pixel operations. Likewise, the software queue 144 of Ben-Yoseph in no way teaches, neither directly nor inherently, a graphics accelerator that works on operands and produces outputs for one set of pixels, while the DMA engine brings in operands for a future set of pixel operations. Applicants point out that claim 29 does not generically claim a FIFO-type software queue. Rather, claim 29 is directed, in part, to a graphics accelerator that works on operands and produces outputs for one set of pixels, while the DMA engine brings in in operands for a future set of pixel operations, which is in no way inherent from the software queue 144 of Ben-Yoseph or buffer 234 of Gulick. As neither Ben-Yoseph nor Gulick disclose this element of amended claim 29, Applicants respectfully request allowance of claim 29.

Claim 30 includes the steps of loading a block of graphics data from main memory into local memory of a graphics accelerator having a processor and a coprocessor, the graphics data including pixels, each pixel having a plurality of components; performing operations on the plurality of components of each pixel of graphics data using the coprocessor; concurrently transferring blocks of unprocessed data and processed data between the main memory and the local memory while the block of graphics data is being processed. There is no disclosure in Ben-Yoseph of the claimed method, including the concurrent transferring of graphics data between local memory and

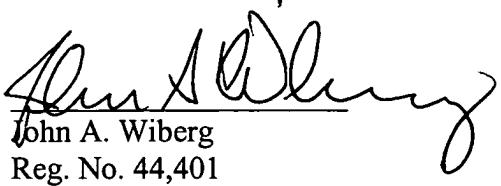
main memory as a block of data is being processed. On page 5 of the Office Action, the Examiner asserts that this method would have been obvious by the teachings of Ben-Yoseph, Col. 3, lns. 40-65. Applicants respectfully disagree with this assertion. Claim 30's concurrent transferring of graphics data between local memory and main memory as a block of data is being processed is made possible by the DMA engine 1304 that has a DMA command queue 1306. Applicants submit that Ben-Yoseph contains no functionality that in itself would perform the functions of claim 30. Accordingly, Applicants respectfully request allowance of claim 30 and all claims dependent thereon.

Claim 61 includes steps of processing a first block of graphics data; transferring a second block of unprocessed graphics data from the main memory to the on-board memory while the first block of graphics data is being processed; transferring a third block of processed graphics data from the on-board memory to the main memory while the first block of graphics data is being processed. There is no disclosure in Ben-Yoseph of the claimed method, including the concurrent transferring of graphics data between local memory and main memory as a block of data is being processed. On page 6 of the Office Action, the Examiner asserts that this method "would have been obvious by the teachings of Ben-Yoseph since Ben-Yoseph teaches simultaneously transmits and receives data over a high speed Rambus DRAM channel, Col. 3, lns. 40-65." Applicants respectfully disagree with this assertion. While Ben-Yoseph teaches simultaneously transmitting and receiving data over the high speed Rambus DRAM channel, it does not disclose the simultaneous processing of graphics data. Claim 61's processing of graphics data while concurrently transferring processed and unprocessed blocks of graphics data between local memory and main memory is made possible by the DMA engine 1304 that has a DMA command queue 1306. Applicants submit that Ben-Yoseph contains no functionality that in itself would perform the functions of claim 61. Accordingly, Applicants respectfully request allowance of claim 61 and all claims dependent thereon.

Based on the foregoing, Applicants submit that claims 1, 2, 5-59, 61 and 62 are in condition for allowance. Applicants, therefore, respectfully request early issuance of a Notice of Allowance.

Date: October 2, 2001

Respectfully submitted,


John A. Wiberg
Reg. No. 44,401

McAndrews, Held & Malloy, Ltd.
500 W. Madison, Suite 3400
Chicago, IL 60661
Ph : 312 775 8000
Fx: 312 775 8100

CUSTOMER NO.



23446

PATENT TRADEMARK OFFICE

MARKED-UP COPY OF PENDING CLAIMS

Cancel claims 4 and 60.

For the Examiner's convenience, all pending claims, both amended and unamended, are included below.

Amend claims 1, 5, 13, 23, 25, 29 and 59 of the application and add new claim 62 as follows:

1. (Twice amended) A graphics accelerator comprising:
[a processor for processing graphics data;]
a [local] memory for storing [the] graphics data, the graphics data including pixels; and
a coprocessor for performing vector operations on a plurality of components of
[one] a pixel of the graphics data, wherein the coprocessor processes the plurality of
components of the pixel in parallel as elements of a vector[; and
a direct memory access engine for transferring the graphics data between an
external memory and the local memory].
2. The graphics accelerator of claim 1 wherein the memory includes a data SRAM.
5. (Amended) The graphics accelerator of claim [4] 1 wherein the plurality of
components of each pixel comprise R, G and B components of RGB formatted graphics
data.
6. The graphics accelerator of claim 5 wherein the pixels are in an RGB16 format.
7. The graphics accelerator of claim 6 wherein the R component has five bits, the G
component has six bits and the B component has 5 bits.
8. The graphics accelerator of claim 6 wherein the graphics data is organized into
32-bit words, and each 32-bit word includes two pixels having RGB16 format.

9. The graphics accelerator of claim 8 wherein the two pixels are respectively selected by two special load instructions.
10. The graphics accelerator of claim 9 wherein the two special load instructions are for loading a left one and a right one of the two pixels, respectively.
11. The graphics accelerator of claim 7 wherein the coprocessor comprises an input register.
12. The graphics accelerator of claim 11 wherein the R, G and B components are expanded into 8-bit components through zero expansion when loaded into the input register.
13. (Amended) The graphics accelerator of claim [4] 1 wherein the plurality of components of each pixel comprise Y, U and V components of YUV formatted graphics data, and wherein the Y, U and V components are also referred to as Y, Cr and Cb components, respectively, of YCrCb formatted graphics data.
14. The graphics accelerator of claim 13 wherein the pixels are in a YUV 4:2:2 format.
15. The graphics accelerator of claim 14 wherein the pixels are organized into 32-bit words and each 32-bit word contains two pixels.
16. The graphics accelerator of claim 15 wherein the two pixels in each 32-bit word is organized in a YUYV format, each of the first Y component, the U component, the second Y component, and the V component occupies eight bits, a first one of the two pixels is comprised of a first Y component, the U component and the V component, and a second one of the two pixels is comprised of the second Y component, the U component and the V component.
17. The graphics accelerator of claim 15 wherein the two pixels are respectively selected by two special load instructions.

18. The graphics accelerator of claim 17 wherein the two special load instructions are for extracting a first one and a second one of the two pixels, respectively.
19. (Amended) The graphics accelerator of claim [4] 1 wherein the coprocessor has an instruction set that includes a special instruction for comparing between each element of a pair of 3-element vectors.
20. The graphics accelerator of claim 19 wherein the coprocessor further comprises a result register, and results of the three comparisons are stored in the result register.
21. The graphics accelerator of claim 20 wherein the results of the three comparisons are used together during a single conditional branch operation.
22. The graphics accelerator of claim 19 wherein the special instruction is for a greater-than-or-equal-to operation.
23. (Twice amended) [The] A graphics accelerator [of claim 3] comprising:
a local memory for storing graphics data, the graphics data including pixels;
a coprocessor for performing operations on a plurality of components of a pixel of
the graphics data; and
a direct memory access (DMA) engine for transferring the graphics data between
an external memory and the local memory, wherein the DMA engine moves data between
the local memory and the external memory [at the same time] while the graphics
accelerator is using the memory for its load and store operations.
24. The graphics accelerator of claim 23 wherein the external memory is a unified memory that is shared by a graphics display system, a CPU and other peripheral devices.
25. (Amended) [The] A graphics accelerator [of claim 3] comprising:
a local memory for storing graphics data, the graphics data including pixels;
a coprocessor for performing operations on a plurality of components of a pixel of
the graphics data; and

a direct memory access (DMA) engine for transferring the graphics data between an external memory and the local memory, wherein the DMA engine includes a queue to hold a plurality of DMA commands.

26. The graphics accelerator of claim 25 wherein the plurality of DMA commands are executed in the order they are received.

27. The graphics accelerator of claim 25 wherein the queue comprises a mechanism that allows the graphics accelerator to determine when all the DMA commands have been completed.

28. The graphics accelerator of claim 25 wherein the queue is four deep for storing up to four DMA commands.

29. (Amended) [The] A graphics accelerator [of claim 3] comprising:
a local memory for storing graphics data, the graphics data including pixels;
a coprocessor for performing operations on a plurality of components of a pixel of the graphics data; and
a direct memory access (DMA) engine for transferring the graphics data between an external memory and the local memory, wherein the graphics accelerator is working on operands and producing outputs for one set of pixels, while the DMA engine is bringing in operands for a future set of pixel operations.

30. (Amended) A method of processing graphics comprising the steps of:
loading a block of graphics data from main memory into local memory of a graphics accelerator having [a processor and] a coprocessor, the graphics data including pixels, each pixel having a plurality of components;
performing operations on the plurality of components of each pixel of graphics data using the coprocessor; and
concurrently transferring blocks of unprocessed data and processed data between the main memory and the local memory while the block of graphics data is being processed.

31. The method of processing graphics of claim 30 wherein the plurality of components comprises R, G and B components of RGB formatted graphics data.
32. The method of processing graphics of claim 31 wherein the pixels of the graphics data are in an RGB16 format.
33. The method of processing graphics of claim 32 further comprising the step of organizing the graphics data into 32-bit words, wherein each 32-bit word includes two pixels having RGB16 format.
34. The method of processing graphics of claim 33 further comprising the step of selecting each of the two pixels with one of two special load instructions.
35. The method of processing graphics of claim 34 wherein the step of selecting each of the two pixels comprises loading a left one of the two pixels.
36. The method of processing graphics of claim 34 wherein the step of selecting each of the two pixels comprises loading a right one of the two pixels.
37. The method of processing graphics of claim 30 wherein each of the plurality of pixels of graphics data comprises Y, U and V components of YUV formatted graphics data.
38. The method of processing graphics of claim 37 wherein the pixels are in a YUV 4:2:2 format.
39. The method of processing graphics of claim 38 further comprising the step of organizing the pixels into 32-bit words, wherein each 32-bit word contains two pixels.
40. The method of processing graphics of claim 39 wherein the step of organizing the pixels into 32-bit words comprises organizing each of the two pixels into a YUYV format, wherein each of the first Y component, the U component, the second Y component and the V component occupies eight bits, a first one of the two pixels is comprised of a first Y component, the U component and the V component, and a second

one of the two pixels is comprised of the second Y component, the U component and the V component.

41. The method of processing graphics of claim 40 further comprising the step of selecting each of the two pixels with one of two special load instructions.
42. The method of processing graphics of claim 41 wherein the step of selecting each of the two pixels comprises loading the first one of the two pixels.
43. The method of processing graphics of claim 42 wherein the step of selecting each of the two pixels comprises loading the second one of the two pixels.
44. The method of processing graphics of claim 30 further comprising the step of comparing between each element of a pair of 3-element vectors, wherein each element of the 3-element vector is one of three components of each pixel.
45. The method of processing graphics of claim 44 wherein the three components of each pixel are R, G and B components.
46. The method of processing graphics of claim 44 wherein the three components of each pixel are Y, U and V components.
47. The method of processing graphics of claim 44 wherein the coprocessor includes a result register, and the method further comprising the step of storing results of the three comparisons in the result register.
48. The method of processing graphics of claim 47 further comprising the step of performing a single conditional branch operation using the results of the three comparisons.
49. The method of processing graphics of claim 44 wherein the step of comparing between each element of a pair of 3-element vectors comprises the step of performing a greater-than-or-equal-to operation between each element of a pair of 3-element vectors.

50. The method of processing graphics of claim 30 wherein the graphics accelerator includes the local memory for loading the graphics data, the method further comprising the step of moving data between the local memory and the external memory using a direct memory access (DMA) engine at the same time the graphics accelerator is using the local memory for its load and store operations.

51. The method of processing graphics of claim 50 wherein the local memory is a data SRAM.

52. The method of processing graphics of claim 50 wherein the external memory is a unified memory that is shared by a graphics display system, a CPU and other peripheral devices.

53. The method of processing graphics of claim 50 wherein the DMA engine includes a queue for holding a plurality of DMA commands.

54. The method of processing graphics of claim 53 further comprising the step of determining whether all the DMA commands have been completed.

55. The method of processing graphics of claim 53 further comprising the step of receiving the plurality of DMA commands.

56. The method of processing graphics of claim 55 further comprising the step of executing the plurality of DMA commands in the order they are received.

57. The method of processing graphics of claim 53 wherein the queue is four deep for storing up to four DMA commands.

58. The method of processing graphics of claim 30 further comprising the step of bringing in operands for a future set of pixel operations using a direct memory access (DMA) engine while the graphics accelerator is working on operands and producing outputs for one set of pixels.

59. (Amended) The graphics accelerator of claim 25 [1, wherein the DMA includes a queue for storing DMA commands,] wherein the [processor] coprocessor posts a plurality

of commands to the queue and the DMA executes the commands concurrently with the operation of the [processor] coprocessor.

61. A method for sequentially processing blocks of graphics data, the method comprising the steps of:

transferring a first block of unprocessed graphics data from main memory to on-board memory;

processing the first block of graphics data;

transferring a second block of unprocessed graphics data from main memory to the on-board memory while the first block of graphics data is being processed; and

transferring a third block of processed graphics data from the on-board memory to the main memory while the first block of graphics data is being processed.

Please add new claim 62 as follows:

Subject D2

-- 62. The graphics accelerator of claim 1 further comprising a direct memory access (DMA) engine for transferring the graphics data between an external memory and the local memory. --